# Data Hiding Steganograph Pada File Image Menggunakan Metode Least Significant Bit

**Dwi Kurnia Basuki, S. Si. M. Kom.[1], Isbat Uzzin Nadhori, S. Kom.[1]**
**Ahmad Mansur Maulana[2]**

[1]Dosen Pembimbing Jurusan Teknik Informatika, [2]Mahasiswa Jurusan Teknik Informatika
Jurusan Teknik Informatika
Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya 60111, Indonesia
Tel: +62 (31) 594 7280; Fax: +62 (31) 594 6114
E-mail : dwiki@eepis-its.edu, isbat@eepis-its.edu, arbalest.mechanical@gmail.com
Homepage : http://www.eepis-its.edu

## *Abstrak*

*Dengan semakin populernya media digital, perhatian pada tingkat keamanan menjadi semakin penting. Salah satu isu penting adalah tingkat keamanan pengiriman suatu informasi. Hal ini dapat dilakukan dengan menggunakan enkripsi atau steganography. Steganography merupakan suatu metode untuk menyisipkan potongan sebuah informasi rahasia dalam suatu objek media lain. Dalam steganography dikenal data hiding atau data embedding yaitu penyembunyian data yang nampak sangat familiar dengan kriptografi. Namun data hiding dalam steganography dan kriptografi sangat berbeda. Jika pada kriptografi, data yang telah disandikan (ciphertext) tetap tersedia, maka dengan steganography, ciphertext dapat disembunyikan sehingga pihak ketiga tidak mengetahui keberadaannya. Dalam proyek akhir ini, media yang diajukan adalah penggunaan media image seperti pada file-file JPG, GIF dan BMP sebagai data masukan media pembawa pesan rahasia (carrier file). Dengan menggunakan metode Least Significant Bit diharapkan nantinya kualitas image yang dihasilkan dari proses embeding data tidak akan terlalu berkurang secara signifikan. Selain itu, untuk lebih merampingkan ukuran data digunakan algoritma zip compression, dan agar menjaga data lebih aman digunakan algortima enkripsi DES. Metode proyek akhir ini membuktikan suatu teknik penyembunyian pesan rahasia dalam media image. Setelah melalui proses embeding data, ouput image hasil dari proyek akhir ini tidak mengalami penurunan kualitas. Begitu juga untuk proses pengekstrakan kembali dari berkas stego, informasi rahasia dapat diungkapkan kembali tanpa mengalami kerusakan sedikitpun.*

**Kata kunci :** kriptografi, data hiding, steganography, image, LSB, zip compression, DES

## 1. Pendahuluan

Keamanan suatu informasi pada jaman global ini makin menjadi sebuah kebutuhan vital dalam berbagai aspek kehidupan[5]. Suatu informasi akan memiliki nilai lebih tinggi apabila menyangkut tentang aspek-aspek keputusan bisnis, keamanan, ataupun kepentingan umum. Dimana informasi-informasi tersebut tentunya akan banyak diminati oleh berbagai pihak yang juga memiliki kepentingan di dalamnya.

Oleh karena itu, *steganography* semakin dibutuhkan guna memberikan keamanan yang maksimal dalam proses pengiriman informasi. *Steganography* merupakan cara untuk menyembunyikan suatu pesan atau data rahasia di dalam data atau pesan lain yang tampak tidak mengandung apa-apa, kecuali bagi orang yang mengerti kuncinya. Teknik *steganography* umum digunakan bersamaan dengan menggunakan dua media yang berbeda, dimana salah satunya berfungsi sebagai media yang berisikan informasi (*carrier file*) dan yang lain berfungsi sebagai media pembawa informasi tersebut (*secret file*).

Melalui penelitian ini dibangun suatu aplikasi berbasis Java yang mengimplementasikan *steganography* dengan menggunakan metode *Least Significant Bit* sebagai cara untuk menyembunyikan suatu data ke dalam media *image*. Penggunaan teknologi *steganography* ini diharapkan dapat membantu upaya dalam peningkatan pengamanan pengiriman informasi dan mempermudah perlindungan atas hak cipta hasil karya media elektronik. Selain itu, untuk lebih merampingkan ukuran data digunakan algoritma *zip compression*, dan agar menjaga data lebih aman digunakan algortima *enkripsi DES*.

## 2. Landasan Teori

### *2.1. Steganografi with LSB (Least Significant Bit)*

Bit atau binary digit adalah unit dasar penyimpanan data di dalam komputer, nilai bit suatu data adalah 0 atau 1. Semua data yang ada pada komputer disimpan ke dalam satuan bit ini, termasuk gambar, suara, ataupun video. Jenis-jenis format pewarnaan di dalam media gambar, seperti grayscale, RGB, dan CMY. Sebagai contoh pewarnaan monochrome bitmap (menggunakan 1 bit

untuk tiap pixelnya), RGB - 24 bit (8 bit untuk Red, 8 bit untuk Green, dan 8 bit untuk Blue), Grayscale-8 bit (menentukan tingkat kehitaman suatu pixel berdasarkan nilai bitnya)[9].

Perhatikan contoh gambar di bawah ini :

*Gambar 1. Contoh gambar*

Gambar di atas menggunakan format pewarnaan grayscale, artinya tiap pixel dari gambar ini direpresentasikan dengan nilai sepanjang 8 bit.

Misalkan sebuah data berupa text "secret", kalau direpresentasikan ke dalam binary kata "secret" ini menjadi :

| Character | ASCII value (decimal) | Hexadecimal | Binary |
|---|---|---|---|
| s | 115 | 73 | 01110011 |
| e | 101 | 65 | 01100101 |
| c | 99 | 63 | 01100011 |
| r | 114 | 72 | 01110010 |
| e | 99 | 63 | 01100011 |
| t | 116 | 74 | 01110100 |

*Tabel 1. Tabel Representasi Kata "secret"*

Sesuai dengan namanya, LSB artinya bit yang tidak significant / tidak mempunyai pengaruh yang besar, maka metode ini mengganti nilai bit ke-8 gambar di atas untuk menyisipkan data. Kalau dipetakan dari kata "secret" akan didapatkan hasil seperti ini (bandingkan dengan table binary sebelumnya) :

Data Binary Media :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00000000 | 00000000 | 00000001 | 00000001 | 00000001 | 00000001 | 00000001 | 00000001 |
| 00000000 | 00000000 | 00000001 | 00000001 | 00000001 | 00000001 | 00000001 | 00000001 |
| 00000000 | 00000000 | 00000001 | 00000001 | 00000001 | 00000001 | 00000001 | 00000001 |
| 00000001 | 00000001 | 00000010 | 00000010 | 00000010 | 00000011 | 00000011 | 00000011 |
| 00000001 | 00000001 | 00000010 | 00000010 | 00000010 | 00000011 | 00000011 | 00000011 |
| 00000001 | 00000001 | 00000010 | 00000010 | 00000010 | 00000011 | 00000011 | 00000011 |

*Tabel 2. Tabel Binary Media*

Data yang ingin disisipkan :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

*Tabel 3. Tabel Binary Data yang akan disisipkan*

Hasil akhir (File Stego) :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00000000 | 00000001 | 00000001 | 00000001 | 00000000 | 00000000 | 00000001 | 00000001 |
| 00000000 | 00000001 | 00000001 | 00000000 | 00000000 | 00000001 | 00000000 | 00000001 |
| 00000000 | 00000001 | 00000001 | 00000000 | 00000000 | 00000000 | 00000001 | 00000001 |
| 00000000 | 00000001 | 00000011 | 00000011 | 00000010 | 00000010 | 00000011 | 00000010 |
| 00000000 | 00000001 | 00000011 | 00000010 | 00000010 | 00000010 | 00000011 | 00000011 |
| 00000000 | 00000001 | 00000011 | 00000011 | 00000010 | 00000011 | 00000010 | 00000010 |

*Tabel 4. Tabel Hasil Berkas Stego*

Setelah diperhatikan, angka yang dibold menunjukkan kalau data tersebut sudah diganti sesuai dengan data yang ingin disisipkan. Beginilah cara metode LSB (Least Significant Bit) bekerja.

### *2.2. Algoritma ZIP Compression*

Algoritma pemampatan data dengan format data ZIP termasuk dalam algoritma kompresi atau pemampatan yang bersifat *lossless*. Berbeda dengan algoritma pemampatan yang bersifat *lossy* yang menghilangkan sebagian informasi dari berkas yang di mampatkan untuk mendapatkan hasil yang optimum, algoritma kompresi yang bersifat lossless seperti ZIP tidak membuang sedikitpun informasi yang dimiliki oleh berkas asal. Algoritma kompresi yang bersifat *lossy* umumnya digunakan untuk memampatkan berkas–berkas gambar, video ataupun suara, hal ini menimbang perubahan (*penghilangan*) sedikit pada berkas asal tidak akan menimbulkan efek yang mampu ditangkap oleh indra manusia. Sedangkan algoritma kompresi yang bersifat *lossless* umumnya digunakan untuk berkas teks atau binary (*executable*). Hal ini mengingat perubahan yang kecil pada berkas yang dikompresi akan memberi pengaruh besar pada berkas hasil kompresi saat di dekompresi ulang. Misalnya pada suatu berkas program computer (*source code*), perubahan yang terjadi walaupun sedikit akan berakibat pada kesalahan kode program tersebut saat di kompilasi setelah di dekompresi.

Berkas termampatkan dengan format zip dibuat dengan menggunakan algoritma kompresi deflate. Sebagaimana format gzip berkas terkompresi dengan format zip dibuat dengan algoritma deflate yang pertama kali didisain oleh Philip Katz (1962-2000), algoritma deflate sendiri merupakan algoritma yang berbasiskan algoritma LZ77 dan kode Huffman (*Huffman Codes*). Spesifikasi format kompresi zip distandardisasi melalui RFC1952 yang ditulis oleh *Peter Deutsch*.

Meskipun *algoritma deflate* tidak dirancang untuk suatu tipe berkas secara spesifik, akan tetapi metode – metode pemampatan data yang dirancang khusus untuk tipe berkas tertentu, yang umumnya memiliki kerumitan yang lebih tinggi, umumnya memiliki performansi (*dalam segi ukuran berkas hasil kompresi*) yang lebih tinggi. Pada umumnya algoritma deflate (*termasuk zip*) memiliki nilai faktor kompresi (*compression factor*) antara 2.5 sampai 3 untuk pemampatan berkas tipe teks dan memiliki nilai yang lebih kecil jika yang berkas dimampatkan adalah tipe binary (*executable*). Factor kompresi (*compression factor*) merupakan invers dari nilai rasio kompresi (*compression ratio*) yang

menunjukkan persentase ukuran berkas hasil pemampatan dibandingkan ukuran berkas sebelum dimampatkan.

$$RK = \frac{Ukuran\_file\_Output}{Ukuran\_File\_Input}$$

RK = rasio kompresi

*Persamaan 1. Rasio Kompresi*

$$FK = \frac{Ukuran\_File\_Input}{Ukuran\_File\_Output}$$

FK = faktor kompresi

*Persamaan 2. Faktor Kompresi*

Dari persamaan tersebut terlihat, bahwa rasio kompresi akan selalu bernilai kurang dari 1, jadi jika suatu algoritma kompresi memiliki nilai rasio kompresi 0,5 maka algoritma ini mampu memampatkan berkas hingga menjadi separuh (50%) dari ukuran semula. Jadi semakin kecil nilai rasio kompresi dari suatu algoritma kompresi maka semakin bagus algoritma tersebut. Berkebalikan dengan nilai rasio kompresi adalah nilai faktor kompresi. Sehingga semakin besar nilai faktor kompresi dari suatu algoritma pemampatan data, maka algoritma tersebut berarti semakin baik. Pada umumnya nilai faktor kompresi lebih sering digunakan sebagai standard ukuran mengingat secara alamiah nilainya menunjukkan tingkat keandalan dari suatu algoritma (semakin besar nilai = semakin bagus kualitas).

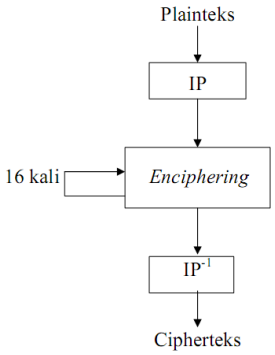
### *2.3. Algoritma DES Encryption*

*DES (Data Encryption Standard)* adalah algoritma *cipher* blok yang populer karena dijadikan standard algoritma enkripsi kunci-simetri, meskipun saat ini standard tersebut telah digantikan dengan algoritma yang baru, *AES*, karena *DES* sudah dianggap tidak aman lagi. Sebenarnya *DES* adalah nama standard enkripsi simetri, nama algoritma enkripsinya sendiri adalah *DEA (Data Encryption Algorithm)*, namun nama *DES* lebih populer daripada *DEA*. Algoritma *DES* dikembangkan di *IBM* dibawah kepemimpinan W.L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma *Lucifer* yang dibuat oleh Horst Feistel. Algoritma ini telah disetujui oleh *National Bureau of Standard (NBS)* setelah penilaian kekuatannya oleh *National Security Agency (NSA)* Amerika Serikat.

*DES* termasuk ke dalam sistem kriptografi simetri dan tergolong jenis *cipher* blok. DES beroperasi pada ukuran blok 64 bit. DES mengenkripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal *(internal key)* atau upa-kunci *(subkey)*. Kunci internal dibangkitkan dari kunci eksternal *(external key)* yang panjangnya 64 bit.

*Cara kerja Algortima DES*

1. Blok plainteks dipermutasi dengan matriks permutasi awal (*initial permutation* atau IP).
2. Hasil permutasi awal kemudian di-enchipering sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan *(invers initial permutation* atau $IP^{-1}$) menjadi blok cipherteks.




*Gambar 2. Skema Global Algoritma DES*

## 3. Perancangan Sistem

Dalam proses ini kita berkonsentrasi pada perancangan desain perangkat lunak, yang meliputi tahap desain data input dan data output, desain proses, dan desain antarmuka.

### 3.1. Rancangan Data

Dalam proses ini kita perlu untuk menentukan bagaimana format data yang akan digunakan dalam proses perancangan dan pembuatan perangkat lunak sistem informasi steganography. Data yang dimaksud merupakan data yang akan digunakan dalam proses input sistem informasi dan data hasil keluaran dari sistem informasi ini.

#### 3.1.1. Data Input

Desain data input yang digunakan pada aplikasi data hiding steganograph ini yang pertama sebagai cover message (file carrier) menggunakan file image berformat GIF, BMP dan JPG. Agar hasil file stego nanti tidak didapatkan hasil yang tidak terlalu jauh berbeda dengan aslinya (file carrier), sebaiknya digunakan file image dengan kedalaman warna 24-bit.

Sedangkan untuk data input yang kedua sebagai secret info (file data) digunakan file dokumen berbagai format dengan ekstensi *.doc, *.xls, *.ppt, *.txt dan juga untuk semua tipe data file *.*

#### 3.1.2. Data Output

Desain data output yang digunakan dalam aplikasi ini didasarkan dari ekstensi cover message (file carrier) pada saat proses input data. Jika pada saat input cover message digunakan file image dengan ekstensi *.jpg maka diharuskan untuk format file outputnya juga menggunakan ekstensi *.jpg. Demikian juga untuk format file image yang lain.

#### 3.1.3. Rancangan Proses

Perancangan proses yang dimaksudkan adalah bagaimana sistem akan bekerja, proses-proses yang digunakan, mulai dari user melakukan input kemudian diproses sampai aplikasi mengeluarkan output berupa stego file.

#### 3.2.1. Pemanfaatan kompresi dan enkripsi untuk meningkatkan kemampuan metode LSB

Steganografi dengan metode LSB (least significant bits) terbukti hanya mampu menyembunyikan informasi yang berukuran minimal, hal ini mendorong dikembangkannya suatu teknik tambahan untuk meningkatkan kemampuan suatu aplikasi steganografi

yang menggunakan metode modifikasi LSB (least significant bits modification).

Steganografi dengan metode LSB hanya mampu menyimpan informasi dengan ukuran yang sangat terbatas. Misalnya suatu citra 24-bit ( R=8 bit, G=8 bit, B=8 bit ) digunakan sebagai wadah untuk menyimpan data berukuran 100 bit, jika masing – masing komponen warnanya (RGB) digunakan satu pixel untuk menyimpan informasi rahasia tersebut, maka setiap pixelnya disimpan 3 bit informasi, sehingga setidaknya dibutuhkan citra wadah berukuran 34 pixel atau setara 34 x 3 x 8 = 816 bit (8 kali lipat ). Jadi suatu citra 24-bit jika digunakan untuk menyimpan informasi rahasia hanya mampu menampung informasi maksimum berukuran 1/8 dari ukuran citra penampung tersebut.

Proses tambahan yang mungkin dilakukan yang dibahas dalam makalah ini adalah dengan melakukan pemampatan data (compression) dengan menggunakan algoritma zip compression. Sehingga informasi rahasia yang akan ditampung ke dalam suatu citra penampung akan berukuran lebih kecil.

Selain itu, untuk meningkatkan keamanan dari metode LSB ini, informasi rahasia yang akan disisipkan ke dalam suatu citra penampung akan dienkripsi terlebih dahulu menggunakan metode DES (Data Encryption Standard)

### 3.2.2. Diagram Alir Sistem

Pembuatan proyek akhir ini merujuk pada alur sistem yang telah dirancang sehingga dapat mengantar pada tujuan yang diharapkan. Secara umum, rancangan skema proses penyisipan informasi dapat digambarkan sebagai berikut :

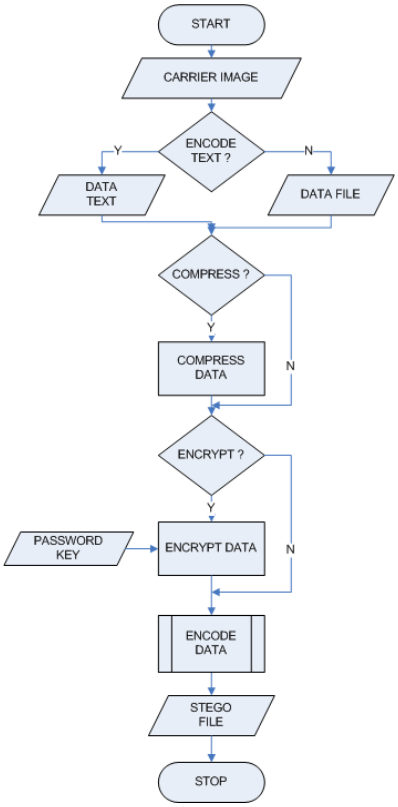



*Diagram 1. Diagram Alir Encoding Data*

Dari diagram alir encoding data di atas, dapat dijelaskan langkah-langkah proses sebagai berikut : setelah memulai sistem (start), selanjutnya user melakukan input untuk *carrier image*. Lakukan pilihan untuk pilihan encoding, apakah data text atau data file. Jika data text, maka akan muncul inputan untuk text, jika data file maka akan keluar menu dialog untuk memilih file mana yang akan disisipkan. Selanjutnya akan muncul pilihan untuk melakukan kompresi dan enkripsi data sebelum melakukan encoding data. Dari proses ini, akan dihasilkan stego file.

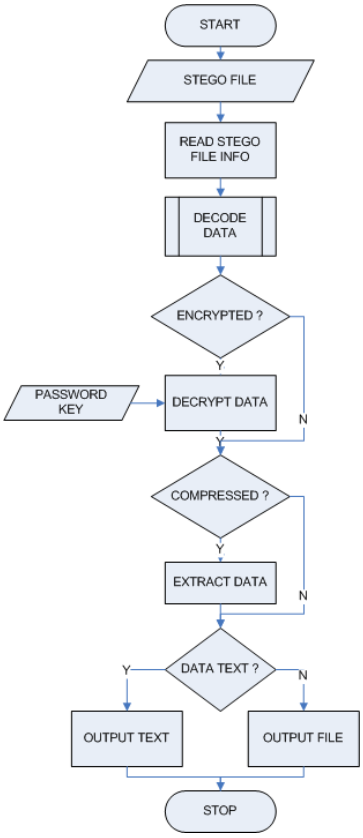


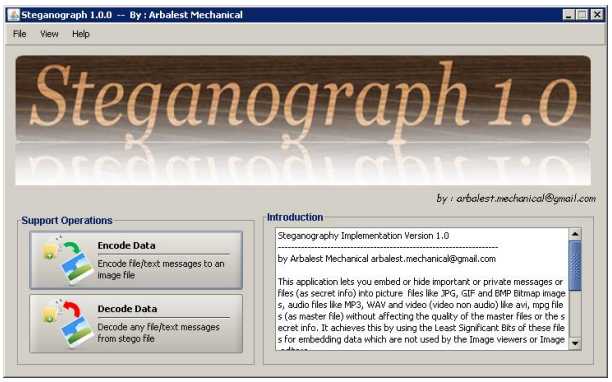

*Diagram 2. Diagram Alir Decoding Data*

Dari diagram alir decoding data di atas, dapat dijelaskan langkah-langkah proses sebagai berikut : setelah memulai sistem (start), selanjutnya user melakukan input untuk stego file. Kemudian dilakukan pembacaan info file stego. Lalu dilakukan encoding data dari file stego. Selanjutnya akan dilakukan pendecryptan dan pengekstrakan data jika pada proses encoding sebelumnya dilakukan encrypt dan compress. Hasil outputnya berupa data teks atau data file tergantung dari informasi yang dibawanya.

### b. Desain Antar Muka

Desain antar muka yang dibuat bertujuan untuk memudahkan user dalam melakukan proses penyisipan dan pengambilan data ke dan dari media gambar.

Dalam desain antar muka ini, penggunaan sistem antar muka dibedakan menjadi 2 bagian utama yaitu bagian untuk menyembunyikan informasi yang nantinya akan disebut encode data dan bagian untuk mengambil informasi dari file stego yang nantinya akan disebut decode data.

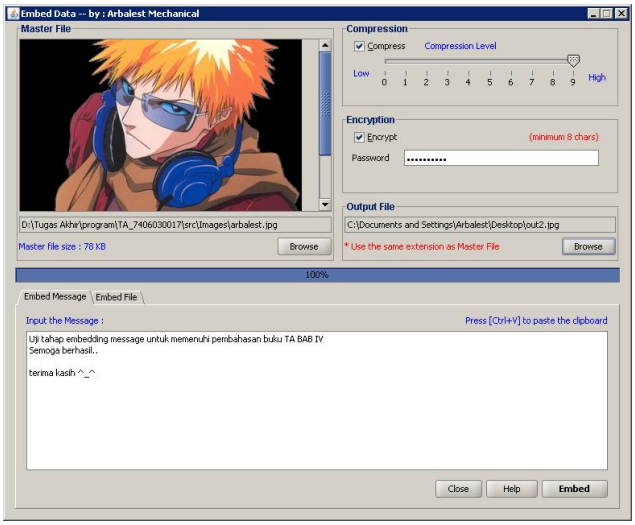



*Gambar 3. Tampilan utama aplikasi*

## 4. Hasil dan Analisa

### 3.1. Proses Embed Message

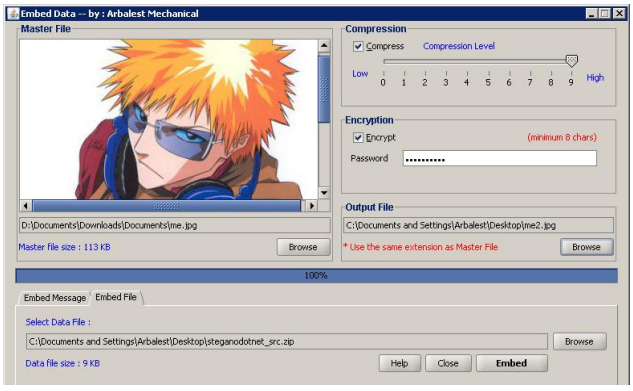
Tampilan window utama untuk melakukan proses embedding message, terlihat juga fitur untuk kompresi yang ditunjukkan dengan slider untuk mengatur level kompresi dan enkripsi data yang ditunjukkan berupa text field untuk menginputkan password. Jika proses embedding message berlangsung sukses maka akan keluar konfirmasi.




*Gambar 4. Tampilan Proses Embedding Message*

### 3.2. Proses Embed File

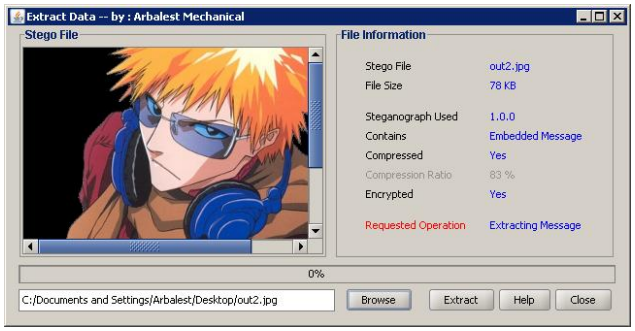
Tampilan window utama untuk melakukan proses embedding file, terlihat juga fitur untuk kompresi dan enkripsi data. Jika proses embedding file berlangsung sukses, maka akan keluar konfirmasi.




*Gambar 5. Tampilan Proses Embedding File*

### 3.3. Proses Retrieve Message

Setelah melakukan browse file berkas stego melalui window utama, maka akan ditampilkan informasi file sebagai berikut :

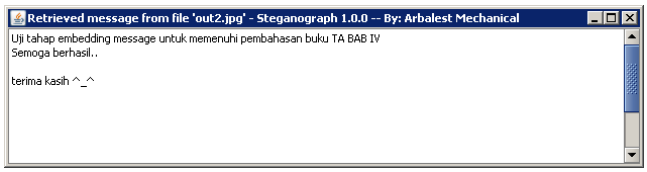



*Gambar 6. Tampilan Proses Ekstrak Message*

Dapat dilihat bahwa, file stego tersebut menyimpan data message, data terkompresi dan terenkripsi. Untuk membukanya, maka dilakukan dengan menekan tombol “Extract” pada menu information sehingga muncul window untuk menginputkan password untuk mendekripsi message


*Gambar 7. Tampilan Input Password*

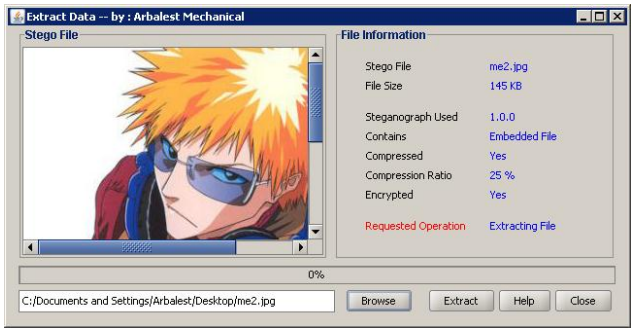
Jika password yang diinputkan benar, maka message yang telah didekripsi tadi akan ditampilkan ke dalam suatu window.




*Gambar 8. Tampilan Output Message*

### 3.4. Proses Retrieve File

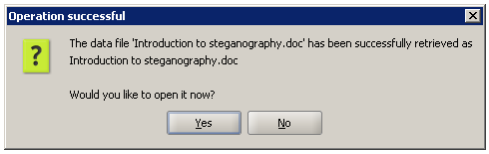
Setelah melakukan browse file berkas stego pada melalui menu utama, maka akan ditampilkan informasi file sebagai berikut :




*Gambar 9. Tampilan Proses Ekstrak File*

Dapat dilihat bahwa, file stego tersebut menyimpan data file, data terkompresi dan terenkripsi. Untuk membukanya, tekan tombol “Extract” sehingga muncul window untuk menginputkan password untuk membuka file data.

Jika password yang diinputkan benar, maka akan muncul dialog konfirmasi apakah file tersebut akan dibuka atau tidak.




*Gambar 10. Tampilan Output Ekstrak File*

### 3.5. Hasil Pengujian

Dalam melakukan pengujian kami memanfaatkan beberapa berkas cover dan beberapa berkas data yang akan disimpan dalam cover. Berkas–berkas yang akan di gunakan tesebut adalah sebagai berikut :

| No | Nama | Ukuran | Tipe |
|---|---|---|---|
| 1 | arbalest1.jpg | 1600x1200px (256 KB) | Berkas JPEG Image 24-bit |
| 2 | mechanical.bmp | 474x474px (660 KB) | Berkas Bitmap Image 24-bit |

*Tabel 5. Berkas Cover*

| No. | Nama | Ukuran | Tipe |
|---|---|---|---|
| 1 | Teks | 20139 karakter | Berkas Teks |
| 2 | Steganography.doc | 124 KB | Berkas Doc Microsoft Word |
| 3 | ultrasurf 9.4.exe | 420 KB | Berkas Binary (Application) |

*Tabel 6. Berkas Data*

| No | Cover | Data | Nama Output | Data | Ukuran Hasil (compressed) | | Stego File Korup |
|---|---|---|---|---|---|---|---|
| | | | | | low | high | |
| 1 | 1 | 1 | 11.jpg | 1 | 275 kB | 262 kB | tidak |
| 2 | 1 | 2 | 12.jpg | 2 | 378 kB | 286 kB | tidak |
| 3 | 1 | 3 | 13.jpg | 3 | 675 kB | 637 kB | tidak |
| 4 | 2 | 1 | 21.bmp | 1 | 680 kB | 667 kB | tidak |
| 5 | 2 | 2 | 22.bmp | 2 | 783 kB | 691 kB | tidak |
| 6 | 2 | 3 | 23.bmp | 3 | 1080 kB | 1042 kB | tidak |

*Tabel 7. Hasil encode data (uncompressed)*

| No | Cover | Data | Ukuran Stego File (compressed) | | Hasil Decode File (corrupted) | |
|---|---|---|---|---|---|---|
| | | | Low | High | Low C | High C |
| 1 | 1 | 1 | 275 kB | 262 kB | tidak | tidak |
| 2 | 1 | 2 | 378 kB | 286 kB | tidak | tidak |
| 3 | 1 | 3 | 675 kB | 637 kB | tidak | tidak |
| 4 | 2 | 1 | 680 kB | 667 kB | tidak | tidak |
| 5 | 2 | 2 | 783 kB | 691 kB | tidak | tidak |
| 6 | 2 | 3 | 1080 kB | 1042 kB | tidak | tidak |

*Tabel 8. Hasil encode data (full compressed)*

## 5. Kesimpulan

Berdasarkan analisa dari beberapa pengujian yang diterangkan pada bagian sebelumnya, kesimpulan yang didapatkan adalah :

1. Aplikasi ini mampu menyisipkan (embed) pesan (message) maupun data file tanpa merusak carrier file ataupun data yang telah disisipkan.
2. Aplikasi ini juga mampu mengambil (retrieve) pesan (message) maupun data file pada berkas file stego.
3. Aplikasi ini menawarkan fitur pengamanan data ganda yaitu dengan melakukan enkripsi terlebih dahulu sebelum disipkan ke dalam carrier file dan juga mampu memperkecil ukuran (compress) data file dengan beberapa level kompresi tertentu.
4. Aplikasi ini dapat memberikan tampilan informasi yang user friendly sehingga memudahkan user untuk mengoperasikannya

## 6. Daftar Pustaka


[1] Al-Mualla, Dr. Muhammed, Al – Ahmad, Prof. Husein, "Information Hiding: Steganography and Watermarking", Etisalat College of Engineering, UAE, 2003

[2] Armyta, Dini, "Makalah Studi Mengenai Aplikasi Steganografi Camouflage Beserta Pemecahan Algoritmanya", Teknik Informatika ITB, 2006

[3] Budi Setiawan, Rachmansyah, "Makalah Penggunaan Kriptografi Dan Steganografi Berdasarkan Kebutuhan dan Karakteristik Keduanya", Teknik Informatika ITB, 2008

[4] Hakim A, Muhammad, "Makalah Studi dan Implementasi Steganografi Metode LSB dengan Preprocessing Kompresi data dan Ekspansi Wadah", Teknik Informatika ITB, 2007

[5] Indoskripsi - Kumpulan Skripsi Online Full Content, "PEMROGRAMAN STEGANOGRAFI DENGAN METODE LSB PADA CITRA DIGITAL", http://one.indoskripsi.com/judul-skripsi/teknik-informatika/pemrograman-steganografi-dengan-metode-lsb-pada-citra-digital
Tanggal akses : 22 Desember 2008 pukul 16.54

[6] Munir, Rinaldi, Diktat Kuliah IF5054 Kriptografi, Teknik Informatika ITB, Bandung, 2005

[7] No name, "Introduction to Steganography", http://io.acad.athabascau.ca/~grizzlie/Comp607/menu.htm
Tanggal akses : 16 Desember 2008 pukul 17.19

[8] Roman Arubusman, Yusrian, "Skripsi - AUDIO STEGANOGRAFI", Universitas Gunadarma, Jakarta, Agustus 2007

[9] Tjong, Andreas, "Steganografi : LSB (Least Significant Bit)", http://andreastjong.wordpress.com/2008/09/22/steganografi-2-lsb-least-significant-bit/
Tanggal akses : 22 Januari 2009 pukul 17.34